ESP8266

The ESP8266 is a **low-cost WiFi module that plays a key role in IoT (Internet of Things) as a WiFi module.** It can be used as a standalone microcontroller or in conjunction with other microcontrollers (such as Arduino) to provide WiFi connectivity.

Main Uses:

- Wireless Networking: The ESP8266 can connect devices to WiFi networks, enabling wireless data transmission and remote control.

In development, if we can transmit data wirelessly without the need for wired connections to a computer, it means we can transmit data anywhere as long as there's a WiFi connection. Besides transmitting data remotely, it can also control devices over the internet, such as smart bulbs.

Using the ESP8266, the DHT11 sensor can collect temperature and humidity data and send it via WiFi to a server (in this case, Python on a computer) for remote monitoring. The same applies to the MAX30102 sensor.

1. Data Collection: **The DHT11 sensor collects temperature and humidity data from the environment.**

2. Data Transmission: **The ESP8266 sends the collected data via WiFi to a server.**

3. Data Processing: **Python on the server receives and processes the data.**

Data Cleaning, Analysis, and Transformation

In recent days, we have collected a large amount of data. We often collected data manually, but during actual data collection, sensors might encounter anomalies or missing data. This leads us to today's topic: **data cleaning, analysis, and transformation**.

Data Cleaning (Handling Missing Values):

Data cleaning involves removing or correcting errors, missing values, and inconsistencies to ensure data quality.

```
import pandas as pd

# Create a dictionary with temperature and humidity data, including some missing values (None)
data = {
    'Temperature': [28.5, 30.1, None, 29.2, 25.7],
    'Humidity': [56, None, 54, 55, 53]
}
```

```
df = pd.DataFrame(data)  # Convert the dictionary to a DataFrame
df_cleaned = df.dropna()  # Remove rows with missing values
df_filled = df.fillna(df.mean())  # Fill missing values with the mean of each
column
print(df)
print(df_cleaned)
print(df_filled)
```

output

```
● PS C:\Users\Frank> & D:/工具文件夹/Python/python.exe c:/Users/Frank/Desktop/u1.py
   Temperature  Humidity
0         28.5      56.0
1         30.1       NaN
2          NaN      54.0
3         29.2      55.0
4         25.7      53.0
   Temperature  Humidity
0         28.5      56.0
3         29.2      55.0
4         25.7      53.0
   Temperature  Humidity
0       28.500      56.0
1       30.100      54.5
2       28.375      54.0
3       29.200      55.0
4       25.700      53.0
```

The df is the clean result. For data cleaning, we handle missing values next. The dropna() function will remove any rows that contain missing values. Then, we fill the missing values with the column's mean using the fillna() function, which can fill in missing values with a specified value. In this case, we fill missing values with the mean of each column.

Data Conversion:

Convert time strings to datetime format for time series analysis. This can be done quickly using Pandas:

```
import pandas as pd

# Create a dictionary with timestamp strings
data = {
    'Timestamp': ['2024-06-21 11:24:32', '2024-06-21 11:24:33', '2024-06-21
11:24:34']
}
df = pd.DataFrame(data)  # Convert the dictionary to a DataFrame
df['Timestamp'] = pd.to_datetime(df['Timestamp'])  # Convert timestamp strings to
datetime objects
print(df)
```

output

```
PS C:\Users\Frank> & D:/工具文件夹/Python/python.exe c:/Users/Frank/Desktop/u1.py
             Timestamp
0 2024-06-21 11:24:32
1 2024-06-21 11:24:33
2 2024-06-21 11:24:34
```

Data Standardization:

Convert Fahrenheit to Celsius and compare recorded Celsius values

```
import pandas as pd

# Create a dictionary with temperature data in Celsius and Fahrenheit
data = {
    'Temp_c': [28.5, 30.1, 29.2, 28.9],
    'Temp_f': [83.3, 86.2, 84.6, 84.0]
}
df = pd.DataFrame(data)  # Convert the dictionary to a DataFrame
df['Temp_f_in_c'] = (df['Temp_f'] - 32) * 5/9  # Convert Fahrenheit to Celsius
print(df)
```

output

```
PS C:\Users\Frank> & D:/工具文件夹/Python/python.exe c:/Users/Frank/Desktop/u1.py
   Temp_c  Temp_f  Temp_f_in_c
0    28.5    83.3    28.500000
1    30.1    86.2    30.111111
2    29.2    84.6    29.222222
3    28.9    84.0    28.888889
```

Conversion formula: Temp_c = (Temp_f - 32) * 5/9

Basic Data Analysis:

Calculate **mean, median, and standard deviation** for temperature and humidity

```
import pandas as pd

# Create a dictionary with temperature and humidity data
data = {
    'Temp_c': [28.5, 30.1, 29.2, 28.9],
    'Temp_f': [83.3, 86.2, 84.6, 84.0],
    'Humidity': [56, 55, 57, 54]
}
df = pd.DataFrame(data)
```

```python
# Calculate mean
temp_mean = df['Temp_c'].mean()
humidity_mean = df['Humidity'].mean()
print('Temp Mean:', temp_mean)
print('Humidity Mean:', humidity_mean)

# Calculate median
temp_median = df['Temp_c'].median()
humidity_median = df['Humidity'].median()
print('Temp Median:', temp_median)
print('Humidity Median:', humidity_median)

# Calculate standard deviation
temp_std = df['Temp_c'].std()
humidity_std = df['Humidity'].std()
print('Temp Std:', temp_std)
print('Humidity Std:', humidity_std)
```

output

```
PS C:\Users\Frank> & D:/工具文件夹/Python/python.exe c:/Users/Frank/Desktop/u1.py
temp_mean = 29.174999999999997
humidity_mean = 55.5
temp_median = 29.049999999999997
humidity_median = 55.5
temp_std = 0.6800735254367729
humidity_std = 1.2909944487358056
```

Data Transformation:

Create new calculated columns and merge columns:

```python
import pandas as pd

# Create a dictionary with temperature and humidity data
data = {
    'Temp_c': [28.5, 30.1, 29.2, 28.9],
    'Temp_f': [83.3, 86.2, 84.6, 84.0],
    'Humidity': [56, 55, 57, 54]
}
df = pd.DataFrame(data)  # Convert the dictionary to a DataFrame
mean_temperature = df['Temp_c'].mean()  # Calculate the mean temperature
df['High_Temp'] = df['Temp_c'] > mean_temperature  # Create a new column
indicating if Temp_c is above the mean
```

```
print(mean_temperature)
print(df)
```

output

```
PS C:\Users\Frank> & D:/工具文件夹/Python/python.exe c:/Users/Frank/Desktop/u1.py
29.174999999999997
   Temp_c  Temp_f  Humidity  High_Temp
0    28.5    83.3        56      False
1    30.1    86.2        55       True
2    29.2    84.6        57       True
3    28.9    84.0        54      False
```

First, import the Pandas library and create a dictionary containing Celsius temperatures, Fahrenheit temperatures, and humidity data. Then, convert the dictionary into a Pandas DataFrame and calculate the mean of the Celsius temperatures. Next, **the code creates a new column named 'High_Temp', which contains boolean values indicating whether each Celsius temperature is above the mean. If the temperature is above the mean, the value is True; otherwise, it is False.**

Handling Large Data:

The real strength of pandas is that pandas has a special function called read excel. By reading excel, we can directly process large amounts of data. Then after we process it, we have to update it to a new excel.

```
import pandas as pd

file_path = '...'  # Define the path to the Excel file
df = pd.read_excel(file_path)  # Read the Excel file into a DataFrame
df = pd.DataFrame(data)  # Create a DataFrame with temperature and humidity data

mean_temperature = df['Temp_c'].mean()  # Calculate the mean temperature
df['High_Temp'] = df['Temp_c'] > mean_temperature  # Create a new column
indicating if Temp_c is above the mean
df.to_excel('...', index=False)  # Save the updated DataFrame to a new Excel file
without the index
```

First, import the Pandas library and define the path to the Excel file. **Then use the pd.read_excel function to read the Excel file from the specified path** and load its contents into a Pandas data frame. **Finally, the code uses the df.to_excel function to save the updated data frame to a new Excel file without the index column.**